IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS

In re application of:	
Runte et al.	Examiner: Qing Chen
Application No: 10/687,233	Art Unit: 2191
Filed: October 15, 2003	
For: TOOLS PROVIDING FOR BACKWARDS COMPATIBLE SOFTWARE	

Mail Stop **Appeal Brief - Patents**
Assistant Commissioner For Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF
IN SUPPORT OF APPELLANT'S APPEAL
TO THE BOARD OF PATENT APPEALS

Applicants (hereinafter "Appellants") hereby submits this Brief in support of an Appeal from a decision of a Final Office Action mailed June 4, 2007, and sustained in an Advisory Action mailed August 14, 2007, for the above-referenced case. Appellants respectfully requests consideration of the accompanying Appeal by the Board of Patent Appeals for allowance of the invention as presently recited in the claims.

FILED VIA EFS WEB November 5, 2007

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST

The real party in interest of the above-referenced U.S. Patent application is SAP AG of Neurottstrasse 16, Walldorf, Germany D-69190, to whom the application has been assigned.

II. RELATED APPEALS AND INTERFERENCES

To the best of Appellants' knowledge, there are no prior or pending appeals, interferences, or judicial proceedings related to the subject matter of this appeal that will directly affect, be directly affected by, or have a bearing on the Board's decision in the pending appeal.

III. STATUS OF CLAIMS

Claims 26-31 and 36-38 have been canceled in the above-referenced application.

Claims 1-25 and 32-35 are pending in the above-referenced application.

Claims 1-25 and 32-35 were finally rejected in the Final Office Action mailed June 4, 2007. These claims are the subject of this Appeal.

IV. STATUS OF AMENDMENTS

In response to the Final Office Action mailed June 4, 2007, Appellants electronically filed a Response After Final on August 1, 2007, which was entered August 1, 2007.

An Advisory Action was mailed August 14, 2007 in response to Appellants' Response of August 1, 2007.

In response to the Advisory Action mailed August 14, 2007, Appellants filed a Notice of Appeal on September 4, 2007, in conjunction with a Pre-Appeal Brief Request for Review.

In response to Appellants' Notice of Appeal and Pre-Appeal Brief Request for Review, a Notice of Panel Decision from Pre-Appeal Brief Review was mailed October 2, 2007.

This Brief is submitted in response to the Panel Decision.

A copy of all claims on appeal is attached hereto as Appendix A.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The claims are summarized as follows. In the summary below, the referenced portion of the Specification should be construed as only representative of the teachings that support the claimed feature(s). Thus, the cited portions are sufficient to support the claim, but are not necessarily the exclusive support in the Specification for such claim features.

1. A computer-implemented method comprising: (Fig. 5; Fig. 6; Fig. 7)

automatically detecting a change introduced into a software object of a first software subsystem, wherein the software object is used by software objects of a second software subsystem; (p. 4, lines 10 to 17; p. 6, line 5 to p. 7, line 17)

determining whether the change is compatible with the software objects of the second software subsystem; and (p. 7, lines 1 to 17)

implementing the introduced change to generate an updated software object if the change is compatible with the software objects of the second software subsystem without introducing any changes into the software objects of the second software subsystem; otherwise, (p. 7, lines 1 to 17)

rejecting the introduced change and generating an error notification. (p. 7, lines 1 to 17)

6. A computer-implemented method comprising: (Fig. 5; Fig. 6; Fig. 7; Fig. 8)

identifying a subset of software objects of a first software subsystem and declaring the subset of software objects frozen; (p. 4, lines 10 to 17; p. 10, line 4 to p. 11, line 10)

detecting a change introduced into a frozen software object from the subset of software objects; and prior to allowing the change, (p. 7, lines 1 to 17)

determining with a compatibility check whether the change is compatible with a second software subsystem; and (p. 7, lines 1 to 17)

issuing a notice indicating results of the compatibility check. (p. 7, lines 1 to 17)

7. The method of claim 6 wherein the subset of software objects declared frozen includes software objects of the first software subsystem that are used by the second software subsystem. (Fig. 2; Fig. 8; p. 4, lines 10 to 17; p. 4, line 23 to p. 5, line 13)

8. The method of claim 7 wherein frozen software objects are classified to include released objects and restricted objects. (Fig. 2; Fig. 8; p. 8, lines 11 to 18)

9. The method of claim 8 wherein the released objects include software objects that are used by the second software subsystem without restrictions. (Fig. 2; Fig. 8; p. 8, lines 11 to 18)

10. The method of claim 8 wherein the restricted objects include software objects that are used by software objects of the second software subsystem, the software objects being fewer in number than a threshold. (Fig. 2; Fig. 8; p. 8, lines 19 to 25)

11. The method of claim 8 wherein an identification of recent changes introduced into a restricted object is provided when software objects of the second software subsystem request new usage of the restricted object. (Fig. 2; Fig. 8; p. 9, line 12 to p. 10, line 2)

12. The method of claim 8 wherein classification of the frozen software objects is based on a number of times a frozen software object is used by the second software subsystem. (Fig. 2; Fig. 8; p. 9, lines 1 to 10)

22. A computer-implemented method comprising: (Fig. 5; Fig. 6; Fig. 7; Fig. 8)

performing a global compatibility check of software objects of a first software subsystem by determining whether any changes were introduced into a subset of the software objects of the first software subsystem since the time of a last compatibility check wherein the introduced changes were introduced without obtaining prior approval; (p. 10, lines 4 to 19)

identifying software objects of a second software subsystem affected by an unapproved change, wherein the affected software objects of the second software system are software objects using at least one software object of the subset of the software objects of the first software system; and (p. 10, lines 4 to 19)

issuing a notice of possible incompatibility between affected software objects and software objects including the unapproved change. (p. 10, lines 4 to 19)

32. An article of manufacture comprising: (Fig. 5; Fig. 6; Fig. 7)

a storage medium having stored therein instructions which, when executed by a processor, cause a processing system to perform a method comprising; (p. 12, lines 6 to 14; p. 12, line 22 to p. 13, line 7)

detecting a change introduced into a software object of a first software subsystem, wherein the software object is used by software objects of a second software subsystem; (p. 4, lines 10-17; p. 6, line 5 to p. 7, line 17)

determining whether the change is compatible with the software objects of the second software subsystem; and (p. 7, lines 1 to 17)

implementing the introduced change to generate an updated software object if the change is compatible with the software objects of the second software subsystem without introducing any changes into the software objects of the second software subsystem; otherwise, (p. 7, lines 1 to 17)

rejecting the introduced change and generating an error notification. (p. 7, lines 1 to 17)

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

CLAIM REJECTIONS UNDER 35 U.S.C. § 101

Claims 26-31 were rejected under 35 U.S.C. § 101 as being directed to non-statutory subject matter. Appellants have elected to cancel these claims, rendering rejection of these claims moot.

CLAIM REJECTIONS UNDER 35 U.S.C. § 112

Claims 26-31 were rejected under 35 U.S.C. § 112, first paragraph, as failing to comply with the written description. Appellants have elected to cancel these claims, rendering rejection of these claims moot.

CLAIM REJECTIONS UNDER 35 U.S.C. § 102

Claims 1-4, 6-20, 22-27, 29-34, 36 and 37 were rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 6,519,767 of Carter et al. (hereinafter "Carter"). As

mentioned above, Appellants have elected to cancel claims 26-27 and 29-31, which renders rejection of these claims moot.

CLAIM REJECTIONS UNDER 35 U.S.C. § 103

Claims 5, 21, 28, 35 and 38 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Carter in view of U.S. Patent No. 6,658,659 of Hiller et al. (hereinafter "Hiller"). Appellants have elected to cancel claim 28, which renders rejection of this claim moot.

VII. ARGUMENT

As mentioned above, Appellants have elected to cancel claims 26-31, which renders rejection of these claims moot. Thus, the rejections of these claims under 35 U.S.C. §§ 101, 112, 102, and 103 will not be addressed herein. The rejections addressed below are the anticipation rejection of claims 1-4, 6-20, 22-25, 32-34, and 36-37 under Carter, and the obviousness rejection of claims 5, 21, 35, and 38 under Carter and Hiller.

The claims do not all stand or fall together, and thus will be argued separately. Specifically, the anticipation rejection under Carter will be argued as follows:

A. Rejection of Claims 1-4 and 32-34

B. Rejection of Claims 6-7 and 13-20

C. Rejection of Claims 8 and 11-12

D. Rejection of Claim 9

E. Rejection of Claim 10

G. Rejection of Claims 22-25

The claims subject to the obviousness rejection under Carter and Hiller are likewise separable under based on the different independent claims from which they depend. Appellants submit that the claims are not unpatentable over the cited references for at least the following reasons.

CLAIM REJECTIONS UNDER 35 U.S.C. § 102

A. Rejection of Claims 1-4 and 32-34

Of the above claims, claims 1 and 32 are independent claims. The limitations of claim 1 are representative of the defects in the Office Action. Claim 1 recites the following:

automatically **detecting a change introduced into a software object of a first software subsystem, wherein the software object is used by software objects of a second software subsystem;**

determining whether the change is compatible with the software objects of the second software subsystem; and

implementing the introduced change to generate an updated software object if the change is compatible with the software objects of the second software subsystem without introducing any changes into the software objects of the second software subsystem; otherwise,

rejecting the introduced change and generating an error notification.

The limitations of claim 32 are directed to an article of manufacture to provide content to carry out such a method.

The Office Actions appear to be hung up on the concept of whether standalone software applications or independent executable files are software objects. Appellants respectfully submit that such an interpretation is outside the scope of the broadest reasonable interpretation that can be provided to the phrase "software object" for a number of reasons. Even assuming for the sake of argument that such an interpretation could be made, the cited reference would still fail to support the rejection.

Carter discusses the compatibility of changes to an interface of an application server. See, e.g., col. 1, line 64 to col. 2, line 23. More particularly, the application server is a standalone software application. Not insignificantly, the compatibility changes are between an original version of the application server with a newer version of the application server. The reference is concerned with making sure the interfaces to the software application are compatible with the previous interfaces. Appellants note that **compatibility of the interfaces** is essential in the Carter reference to make sure that the object servers (applications) are compatible with other applications. Carter defines incompatibility such that an update or modification to an **object server** causes the object server to **not support the same interfaces** as its predecessor object server. See col. 3, lines 29 to 41. The "preferred embodiment" of the so-called "compatibility analyzer" as described at col. 13, line 47 to col. 14, line 16, is explained as: "In the preferred embodiment, version compatibility analyzer 70 performs this comparison by comparing the type information of each class in new object server 64 to the type information of [an] identically named class in existing object server's type library 150." Appellants note the significant absence of any detecting of a change to a **software object**, in contrast to what is recited in Appellants' claims. As appears from the cited reference, Carter fails to consider changes at all to a software

object, and is rather concerned with changes to interfaces to the **applications** (specifically, the object servers).

The interpretation of "software objects" as used in Appellants' claims is apparently being interpreted by the Office as being broad enough to encompass executable files or standalone applications, such as Carter's object servers. See more particularly, the Final Office Action at page 26, which asserts that because the object server is an "executable file," it is a program, and thus a software object. The assertion is repeated in the Advisory Action at the Continuation page, stating:

> the executable file of Carter et al. can be interpreted, under the broadest reasonable treatment, as a software object in accordance with the exemplary definition of a software object provided by the originally-filed specification as to include programs. One of ordinary skill in the art would clearly associate an executable file as a program."

Appellants submit that the assertion in the Office Actions represents a misapplication of the standards for examination. Per MPEP § 2111, "During patent examination, the pending claims must be 'given their **broadest reasonable interpretation consistent with the specification**.'" Citations omitted, emphasis added. This principle was violated in the Office Actions because the interpretation used to reject the claims is inconsistent with the specification, as Appellants have previously argued. Appellants repeat that the reasoning in the Final Office Action is contrary to what would be understood by one of skill in the art.

As a first matter, such a broad interpretation of the expression "software objects" is contrary to the way that term is used by the reference itself. Carter at col. 1, lines 17 to 25 and 33 to 43 recites the following:

> **The term "object" generally refers to an instance of a programmer-defined data structure (the data or variables of which are referred to as the object's "properties") and functions that manipulate that structure** (referred to as "member functions," "member procedures," or simply "members" of the object). In other words, **objects are a combination of a data structure with a set of functions that perform methods on the data structure**. The term "class" generally refers to the definition of an object....
>
> One benefit to the use of objects is that their members define a standard way for other programs to interact with or access the object's properties. A set of semantically related functions implemented on an object is generally referred to as an "interface" of the object. Each object typically includes at least one interface, but in some object-oriented systems (such as OLE 2 described below) may include more than one interface. **By allowing access to an object's**

members through an interface, the object effectively "exposes" its functionality for use by other application programs....

As shown in the reference itself, the term object is defined in such a way that a standalone application is excluded from its definition. In fact, objects and "application programs" are discussed as being on conceptually different levels, as one of skill in the art would understand them to be. Appellants submit that interpreting "software objects" to include application programs would be inconsistent with the definitions provided in Carter itself.

Even assuming the definitions in Carter were improperly ignored, Appellants submit that the expression "software object" cannot reasonably interpreted to include application programs at least based on the understanding of those skilled in the art. Consider, for example, the fact that interfaces to application programs are not the same as interfaces to software objects. The comparison of compatibility of the interfaces to an application does not necessarily or inherently disclose compatibility of interfaces to a software object. Even assuming the comparison of compatibility of the interfaces to an application does disclose compatibility of interfaces to a software object, the comparison of compatibility of interfaces to an application program does not necessarily or inherently disclose comparing whether a change to an object of one software subsystem is compatible with a second subsystem.

Furthermore, assuming the application programs of Carter are somehow interpreted as software objects, the Office has failed to point to what is purported to be the "subsystems" as claimed. A claim rejection is inadequate when it fails to show the same detail as what is claimed. See MPEP 706. The object servers are different versions of each other running on the same software environment. There is no assertion of different software subsystems. If the object server is the object, it cannot be the subsystem, because then it would be an object within itself as both the subsystem and object. Thus, there is a logical inconsistency with the interpretation in the Office Actions.

Furthermore, the Final Office Action makes reference to paragraph [0016] of Appellants' Description, which states: "The term 'object', as used herein, means a software object including, but not limited to, function modules, programs, data objects, classes, class components, attributes, etc." Thus, Appellants' disclosure refers to an object as a "program." Carter refers to "executable files" as "programs." Appellants note that one of skill in the art would appreciate that there are two separate meanings for the term "program." One definition is consistent with Carter,

which is that an executable file (an application) is sometimes referred to as a program. However, a program does not necessarily mean an application program, and thus the term "program" can have at least one meaning not consistent with Carter.

For example, a second, distinct definition of a program is referred to in Appellants' paragraph [0016], referring to a function module, routine, or subroutine callable in an application (not being the application itself), to perform a certain function, algorithm, method, etc. The second definition is consistent with Appellants' Specification, which refers to programs in the same context as "classes, class components, [and] attributes." The first definition would be inconsistent with a discussion of "classes, class components, [and] attributes." The term program in Appellants' Specification appears only in paragraph [0016], and the claims. There is no evidence to assume that program means an application program as referenced in Appellants' Specification.

Appellants submit that the reasoning in the Office Action equating "program" to the claimed "objects" is similar to the following hypothetical situation. Consider a patent application that made reference to "pointing devices such as a mouse, a trackball, or a touchpad." Then consider that a hypothetical Office Action cited a reference that referred to a rodent, and specifically a mouse. The hypothetical Office Action may proffer that one of skill in the art would understand that a mouse can refer to a pointing device, which means the rodent discloses the pointing device. Appellants emphasize that the mere fact that the terms ("mouse") are identical on their face does not necessarily mean they have the same meaning. While this example may be slightly exaggerated, the Office Action in the present case has similarly, improperly equated a term (program, meaning executable file) with another term (software object, which may include a type of program or function module) that is not a direct equivalent. Thus, Appellants respectfully submit that the Carter reference fails to support the rejection in the Office Actions. The reasoning of the Final Office Action is not logical, and the application of the Carter reference to the claims is misplaced.

Furthermore, Appellants submit that such effort by the Office in attempting to define "objects" as including application programs misses the point of the claims. The Final Office Action also fails to provide a prima facie case of anticipation under MPEP § 2131, for failing to make a rejection of at least certain features of the claimed invention. As Appellants stated in a previous Response, "Without conceding that Carter's application servers are the same as the

objects recited in the independent claims, Appellants submit that the rejection is defective. The Final Office Action fails to make any assertion that objects of different subsystems are disclosed in the cited reference." The Advisory Action fails to address this defect of the rejection, and fails to address Appellants' argument. Thus, the Advisory Action is incomplete and non-responsive, and the arguments made by Appellant have not all been addressed by the Office, denying Appellants the opportunity to be heard. Appellants reiterate that the claims recite first and second subsystems, whereas Carter is only concerned with compatibility of application servers with a single system. The reference therefore fails to support a rejection of the invention as recited in Appellants' claims.

The remaining claims, as dependent claims, are necessarily patentable over the cited reference for at least the same reasons set forth above with respect to claims 1 and 22.

B. Rejection of Claims 6-7 and 13-20

As a first matter, independent claim 6 was rejected on the same basis as independent claims 1 and 22 discussed above. For at least the same reasons as discussed above, Appellants submit that independent claim 6 and its dependents are patentable over Carter. That is, the arguments made above in sub-section "A" apply equally well to these claims.

Furthermore, claim 6 recites other features not discussed above. Claim 6 recites the following:

> **identifying a subset of software objects of a first software subsystem and declaring the subset of software objects frozen;**
> detecting a change introduced into a frozen software object from the subset of software objects; and prior to allowing the change,
> determining with a compatibility check whether the change is compatible with a second software subsystem; and
> issuing a notice indicating results of the compatibility check.

Thus, Appellants' claim refers to frozen software objects that are software objects of a first software subsystem used by a second software subsystem. The Office Actions attempt to reject such features under col. 13, line 47 to col. 14, line 32 of Carter. That section of Carter describes comparing identically named classes of the updated object server with the previous version of the object server. Thus, there are no "frozen objects" as claimed, which are objects identified as software objects of a first software subsystem used in a second subsystem. See also, paragraph [0016] of Appellants' Specification. Appellants note that this rejection is one more instance of

the Office Action mixing and matching its rejections to match Appellants' claims, rather than making and sticking to a consistent, broadest reasonable interpretation of terms. In this case, the Office refers to the object servers as the subsystems themselves, where in the rejection of the independent claims, the Office declared the object servers to be "software objects." Thus, Appellants submit that the rejection in the Office Actions is defective at least for being self-inconsistent.

Claim 7 is dependent on claim 6 and similarly recites frozen software objects. Thus, the arguments made above with respect to claim 6 apply equally well to dependent claim 7.

C. Rejection of Claims 8 and 11-12

These claims depend from claim 6, discussed above. All arguments made above with respect to claim 6 apply equally well to these claims. Furthermore, these claims recite limitations directed to "released objects" and "restricted objects." The discussion above with respect to the rejection of claims 6 and 7 is similar to the line of reasoning proffered by the Office with respect to these claims. That is, the Office recites different versions of elements of the object server as showing the released and restricted objects. The argument in the Office Action is not consistent with its declaration that the object servers are the software objects. Furthermore, even assuming the reference discloses software objects, the Office has failed to assert frozen software objects classified as released and restricted objects. Thus, the rejection is incomplete at best, and Appellants submit that it is defective for failing to show the claimed features with the specificity claimed.

D. Rejection of Claim 9

The discussions with respect to subsections A through C apply equally well to claim 9. Claim 9 further recites limitations directed to released objects including software objects used by the second software subsystem without restrictions. The Office merely points to the same section of the Carter reference used to reject claims 6, 7, and 8. The Office makes no attempts to provide reasoning to explain how the classes of the object servers are supposedly used with or without restrictions. The Office fails to establish that such a feature is either expressly or inherently present in the cited reference. Thus, the Office Action fails to establish a prima facie case of anticipation with respect to this claim.

E. Rejection of Claim 10

The discussions with respect to subsections A through C apply equally well to claim 10. Claim 9 further recites limitations directed to restricted objects including software objects used by the second software subsystem fewer than a threshold number of times, or a number of the objects being lower than a threshold (i.e., a number of instances). The Office merely points to the same section of the Carter reference used to reject claims 6, 7, and 8. The Office makes no attempts to provide reasoning to explain how the classes of the object servers are supposedly restricted in numbers of instances or not. The Office fails to establish that such a feature is either expressly or inherently present in the cited reference. Thus, the Office Action fails to establish a prima facie case of anticipation with respect to this claim.

F. Rejection of Claims 22-25

Of these claims, claim 22 is independent, and recites the following:

> performing a global compatibility check of software objects of a first software subsystem by determining **whether any changes were introduced into a subset of the software objects of the first software subsystem since the time of a last compatibility check** wherein the introduced changes were introduced without obtaining prior approval;
>
> identifying software objects of a second software subsystem affected by an unapproved change, wherein the affected software objects of the second software system are software objects using at least one software object of the subset of the software objects of the first software system; and
>
> issuing a notice of possible incompatibility between affected software objects and software objects including the unapproved change.

Thus, in addition to reciting limitations directed to software objects and a compatibility check of software objects of a first software subsystem with a second software subsystem, claim 22 recites limitations directed to whether changes were introduced since the time of a last compatibility check. Although the difference may be somewhat subtle, Appellants submit that such a feature is not disclosed, even under the most liberal interpretation of the cited reference.

As a first matter, Appellants submit that the reasoning provided above in subsection A with respect to claim 1 applies equally well to claim 22 and its dependents. Furthermore, regarding the other feature pointed out, Appellants note that claim 22 implies that changes may be made without a compatibility check being performed, or rather, that the timing of a

compatibility check and the timing of a change to a software object do not necessarily coincide. Such a feature is absent in the cited reference, which requires a compatibility check each time the application program is changed. Such a requirement in the cited reference makes sense when considering the fact that the object servers are application programs that require compatibility prior to deployment, rather than being actual software objects. Thus, Appellants submit that the independent claim recites features not disclosed or suggested in the cited reference. Therefore, the cited reference fails to support an anticipation rejection of either claim 22 or its dependent claims.

CLAIM REJECTIONS UNDER 35 U.S.C. § 103

Rejection of claims 5, 21, 28, 35 and 38 under Carter and Hiller

Claims 5, 21, 28, 35 and 38 were rejected under 35 U.S.C. § 102(e) as being unpatentable over Carter in view of U.S. Patent No. 6,658,659 of Hiller et al. (hereinafter "Hiller"). Appellants have elected to cancel claim 28, which renders rejection of this claim moot. Furthermore, claims 5, 35, and 38 depend from claims 1 and 32, discussed in subsection A above. Claim 21 depends from claim 6 discussed in subsection B above. The arguments provided in those subsections apply equally well to the dependent claims. With respect to the obviousness rejection of these claims, Appellants note that the combination of Carter and Hiller must disclose or suggest every feature of the invention as recited in the independent claims to support a prima facie case of obviousness for the claims that depend from those independent claims. The references do not support a prima facie case of obviousness of the independent claims.

Hiller is not cited as curing the deficiencies of Carter discussed above. Hiller discusses a "version aware" loading module that loads only software modules that are compatible with each other. However, Appellants submit that like Carter, Hiller fails to disclose or suggest comparing whether a change to an object of one software subsystem is compatible with a second subsystem, as discussed above in subsection A, or identifying a subset of software objects of a first software subsystem and declaring the subset of software objects frozen as discussed above in subsection B. Thus, Hiller suffers at least the same defects as Carter, and so fails to support a rejection of the independent claims. For at least these reasons, the combination of Carter and Hiller fail also to disclose or suggest the invention as recited in these dependent claims.

In conclusion, Appellants respectfully submit that all appealed claims in this application are patentable and requests that the Board of Patent Appeals and Interferences overrule the Examiner and direct allowance of the rejected claims.

A single copy of this correct brief is submitted as per 37 C.F.R. §41.37(a). Appellant believes that no fee is required, as the fee of $500.00 to cover the appeal fee for one other than a small entity as specified in 37 C.F.R. §1.17(c) was submitted with the originally filed Brief. Please charge any shortages and credit any overcharges to our Deposit Account No. 02-2666.

Respectfully submitted,
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN, LLP

Date: November 5, 2007

/Vincent H. Anderson/
Vincent H. Anderson
Reg. No. 54,962
Attorney for Appellant

1279 Oakmead Parkway
Sunnyvale, CA 94085-4040
(503) 439-8778

VIII. CLAIMS APPENDIX

1. A computer-implemented method comprising:

automatically detecting a change introduced into a software object of a first software subsystem, wherein the software object is used by software objects of a second software subsystem;

determining whether the change is compatible with the software objects of the second software subsystem; and

implementing the introduced change to generate an updated software object if the change is compatible with the software objects of the second software subsystem without introducing any changes into the software objects of the second software subsystem; otherwise,

rejecting the introduced change and generating an error notification.

2. The method of claim 1 wherein determining whether the change is compatible further comprises determining whether the change is predefined as compatible.

3. The method of claim 2 further comprising allowing the change if the change is predefined as compatible.

4. The method of claim 3 further comprising issuing a message that the change is not allowed if the change is not predefined as compatible.

5. The method of claim 4 further comprising allowing the change if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible.

6. A computer-implemented method comprising:

identifying a subset of software objects of a first software subsystem and declaring the subset of software objects frozen;

detecting a change introduced into a frozen software object from the subset of software objects; and prior to allowing the change,

determining with a compatibility check whether the change is compatible with a second software subsystem; and

issuing a notice indicating results of the compatibility check.

7. The method of claim 6 wherein the subset of software objects declared frozen includes software objects of the first software subsystem that are used by the second software subsystem.

8. The method of claim 7 wherein frozen software objects are classified to include released objects and restricted objects.

9. The method of claim 8 wherein the released objects include software objects that are used by the second software subsystem without restrictions.

10. The method of claim 8 wherein the restricted objects include software objects that are used by software objects of the second software subsystem, the software objects being fewer in number than a threshold.

11. The method of claim 8 wherein an identification of recent changes introduced into a restricted object is provided when software objects of the second software subsystem request new usage of the restricted object.

12. The method of claim 8 wherein classification of the frozen software objects is based on a number of times a frozen software object is used by the second software subsystem.

13. The method of claim 6 wherein a software object is a function module.

14. The method of claim 6 wherein a software object is a data structure.

15. The method of claim 13 wherein the software object includes an environment of the function module.

16. The method of claim 6 wherein a software object includes a class and an environment of the class.

17. The method of claim 6 wherein a software object includes an interface and an environment of the interface.

18. The method of claim 6 wherein a software object includes a program and an environment of the program.

19. The method of claim 6 wherein the detecting the change comprises automatically monitoring development of software code.

20. The method of claim 6 wherein the determining whether the change is compatible comprises determining whether there is a predefined declaration of compatibility of the change.

21. The method of claim 7 wherein the determining whether the change is compatible comprises determining whether an expert declared the change compatible.

22. A computer-implemented method comprising:

performing a global compatibility check of software objects of a first software subsystem by determining whether any changes were introduced into a subset of the software objects of the first software subsystem since the time of a last compatibility check wherein the introduced changes were introduced without obtaining prior approval;

identifying software objects of a second software subsystem affected by an unapproved change, wherein the affected software objects of the second software system are software objects using at least one software object of the subset of the software objects of the first software system; and

issuing a notice of possible incompatibility between affected software objects and software objects including the unapproved change.

23. The method of claim 22 wherein the performing a global compatibility check comprises comparing a current version of software code with a version of the software code at the time of a last global compatibility check.

24. The method of claim 22 wherein the subset of the software objects includes frozen software objects.

25. The method of claim 24 wherein the frozen software objects include software objects of the first software subsystem used by software objects of the second software subsystem.

26-31. (Canceled)

32. An article of manufacture comprising:

a storage medium having stored therein instructions which, when executed by a processor, cause a processing system to perform a method comprising;

detecting a change introduced into a software object of a first software subsystem, wherein the software object is used by software objects of a second software subsystem;

determining whether the change is compatible with the software objects of the second software subsystem; and

implementing the introduced change to generate an updated software object if the change is compatible with the software objects of the second software subsystem without introducing any changes into the software objects of the second software subsystem; otherwise,

rejecting the introduced change and generating an error notification.

33. The article of manufacture of claim 32 wherein the instructions, which when executed by the processor, cause the processing system to perform the method wherein determining whether the change is compatible further comprises determining whether the change is predefined as compatible.

34. The article of manufacture of claim 32 wherein the instructions, which when executed by the processor, cause the processing system to perform the method further comprising issuing a notification that the change is not allowed if the change is not predefined as compatible.

35. The article of manufacture of claim 32 wherein the instructions, which when executed by the processor, cause the processing system to perform the method further comprising allowing the change if an expert declares the change compatible upon receiving a request for a manual compatibility check, wherein the change is not predefined as compatible.

36-38. (Canceled)

IX. EVIDENCE APPENDIX

None.

X. RELATED PROCEEDINGS APPENDIX

None.